Error Messages C

This appendix is a guide to the error messages THINK C, THINK Rez, and the THINK Project Manager can generate. Error messages generated by Symantec C++ are documented in the *Symantec C++ Compiler Guide.*

When the compiler comes to something in your source code that it doesn't understand, it prints the file name and line number:

```
file "trash.c"; Line 6
Error: undefined identifier ‘a’
file "trash.c"; Line 8
Error: no return value for function ‘main’
```

Underneath is either an error or warning message explaining why the compiler couldn't understand the line. All possible messages are described in this appendix.

The following internal errors should never occur:

internal error (AER)
internal error (FF2)
internal error (MPS)
internal error (CER)

If you see one, quit the THINK Project Manager and restart your computer. Then call Symantec Technical Support using the instructions in the section "Technical Support" in the "Symantec Customer Service Plan" at the end of this guide.

internal error (ZREF)
You may see this error when your project has become corrupted. To learn how to recover your project, see "Recovering Corrupted Projects" in Chapter 8, "Organizing Projects."

Error Messages

Descriptions of error messages marked *Assembly* are generated by the THINK C inline assembler. Error messages marked *Debugger* are generated by the THINK C Debugger. Error messages marked *THINK Rez* are generated by THINK Rez. Messages marked *Warning* point out statements that are legal code but that may not execute as you expect.

Code that generates only warning messages (and no error messages) will compile.

The error messages in this appendix are broken into the following catagories:

1. Error messages that begin with numbers
2. Error messages that begin with '*metaname*'
3. Error messages that begin with a pound sign (#)
4. Error messages that begin with dollars signs ($$)
5. All remaining error and warning messages in alphabetical order

8-bit reference to label *'symbol'* out of range
Assembly. You've used a symbol that's out of range for an assembly language instruction that expects an 8-bit operand. This error occurs if you try to do a short branch to a label that's over 127 bytes away from the branch instruction, or use such a label in a PC-relative with index addressing mode. Because the inline assembler optimizes branches to `goto` statements, you'll also get this error if the final destination of the branch is over 127 bytes away. Example:

```
asm {
        bar.s @foo
        jmp   @foo(d0.w)
        …  // over 127 bytes
foo:

kansas:  bar.s @cLabel
}

cLabel:  goto Oz
         …  // over 127 bytes from kansas
Oz:      printf( "We're not in "
                 "kansas anymore!" );
```

Keep in mind that `return`, `continue`, and `break` are treated just like `goto` in the inline assembler.

0 or 1 expected

THINK Rez. After the characters `0b`, you can use only binary digits. There should be no spaces between the `b` and the number.

"filename" is not a text file

You get this error by doing one of the following:

- Including more than one precompiled header.
- Referring a file name in an `#include` statement that is not a text file.
- Option-double-clicking on a symbol to find its definition, when the symbol is defined in a library.

'identifier' is already defined

You've already declared the item.

'symbol' has not been declared

You used the identifier *symbol* before you declared it.

'symbol' is not an argument

You declared *symbol* as if it was an argument in a function definition, but it is not in the parameter list. Example:

```
void function(void)
short left_out_of_parameter_list; //NO
{
}
```

'*symbol*' was not placed in a register

Assembly. THINK C couldn't place a variable in a register, and you tried to use the variable as a register in an inline assembly language section. Example:

```
void function(void)
{
   register short a, b, c, d, e, f;

   asm {
      moveq #10,f    ;f isn't in a register
   }
}
```

#printf statements can't have more than 20 arguments

THINK Rez. THINK Rez won't compile any #`printf` statement with more than 20 arguments.

#printf statements do not end with a ';'

THINK Rez. Unlike C, in which the `printf` statement is a function call, THINK Rez considers #`printf` a preprocessor statement, similar to a #`define`.

$$resource statement failed

THINK Rez. A `$$resource` statement fails if the source file can't be found, or if the given resource type, ID or name is invalid.

$$shell is not supported

THINK Rez. The THINK Project Manager does not support exported shell variables, so THINK Rez will not compile files that contain any `$$shell` statements.

access to '*symbol*' denied

The member *symbol* of a class is `private` or `protected`, or was inherited `private`, and it is not accessible. Remember, if you

declare a class with the `class` keyword, its members and inheritance are `private` by default. Example:

```
class A {
   short i;
   void foo(void);
};

class B : A {
public:
   short j;
   void bar(void);
};

class C : B {
public:
   short k;
   void xyzzy(void);
};

class D : public B {
public:
   short k;
   void xyzzy(void);
};

void function(void)
{
   A *p = new A;
   B *q = new B;
   C *r = new C;
   D *s = new D;

   p->i = 0;  // NO: i is private
   q->j = 0;  // OK: j is public
   r->j = 0;  // NO: j is inherited private
   s->j = 0;  // OK: j is inherited public
}
```

access to 'delete' denied

You tried to `delete` an object, and its class defines its destructor as private or protected. All members of a class declared with the `class` keyword are `private` if no access specifier is provided. You might

declare constructors and destructors `private` to control allocation and deallocation through `static` member functions. Example:

```
class ClassC {
public:
   static void destroy(ClassC *);
private:
   ClassB   *b;
   ~ClassC(void);
};

void ClassC::~ClassC(void)
{
   delete b;
}

void ClassC::destroy(ClassC *c)
{
   delete c;      // OK.
}

void function(void)
{
   register C *p;

   delete p;      // NO: Destructor is
                  //     private.
   C::destroy(p); // OK.
}
```

access to 'new' denied

You tried to create an object with `new`, and its class defines its constructor as `private` or `protected`. Note that all members of a class declared with the `class` keyword are `private` if no access specifier is provided. You can declare constructors and destructors

`private` to control allocation and deallocation through `static` member functions. Example:

```
class ClassC {
public:
   static ClassC *make(void);
private:
   ClassB   *b
   ClassC();
};

ClassC::ClassC(void)
{
   b = new ClassB
}

ClassC *ClassC::make(void)
{
   return(new ClassC);  // OK.
}

void function(void)
{
   register ClassC *p;

   p = new ClassC;      // NO: Constructor
                        //     is private.
   p = ClassC::make();  // OK.
}
```

addressing mode requires 68020

For more information on using the inline assembler, see Chapter 20, "Writing Inline Assembly."

Assembly. You're trying to use an MC68020 addressing mode, when MC68020 inline assembly is not enabled. Example:

```
asm {
   clr.w (100000,a0)    ;NO
   clr.w (10000,a0)     ;OK: Optimized to
                        ;    10000(a0)
   clr.w (1000,a0,d0.w) ; NO
   clr.w (100,a0,d0.w)  ; OK: Optimized to
                        ;     100(a0,d0.w)
   clr.w 100(a0,d0.w*2) ; NO: Scale factor
}                       ;     must be 1.
```

AppleEvent timed out—server may still be processing

You usually get this message when you try to perform an operation while ToolServer or SourceServer is still processing a request from THINK Project Manager. Some requests can take several minutes to fulfill. THINK Project Manager stops waiting for a response after

about a minute (as a fail-safe in case the server is hung), but the server continues processing the request until it has finished.

argument list is inappropriate here

You declared a function with an old-style argument list without providing a function body. Usually, you specify a prototype for a function using a type name that is undefined. Example:

```
short foo(Str255);   // NO: If Str255 isn't
                     //     defined.

#include <Types.h>
short foo(Str255);   // OK
```

argument to function does not match prototype
argument to function *'symbol'* does not match prototype

The sole argument to a function does not match the prototype. The prototype may be explicit or inferred. Example:

```
void print(char *);
void function(void)
{
   char *msg = "hello, world\n";
   Str255 s;

   print(s);   // NO: Str255 is
               //     'unsigned char'
   print(msg); // OK.
   error(s);   // OK: Infers
               // int error(unsigned char *)
   error(msg); // NO: Doesn't match
               //     inferred prototype.
}
```

argument #*n* to function does not match prototype
argument #*n* to function *'symbol'* does not match prototype

The given argument to a function does not match the prototype. The prototype may be explicit or inferred.

array nesting too complex

THINK Rez. THINK Rez will not compile a nested array that is more than 20 levels deep.

attempt to initialize an array in a type statement

THINK Rez. The only kind of arrays that can be initialized when declared are `rect` and `point`. All other arrays must be defined as a resource statement.

```
type 'test' {
   rect = { 0, 0, 100, 100 };
                        //valid type statement
   point = { 10, 10 };//valid type statment
   integer = { 1, 2 };//not a valid array
                        //type
   array {
      integer;          //valid, array elems
                        //defined in resource
   };
};
```

bad constant value

THINK Rez. Some statements require a constant value. Make sure that your statement is not attempting to use a computed value when a constant value is required. For example, an enumerated value must be a constant value.

```
type 'test' {
   short x = unknown_label_value;
         /* must be a constant value */
};
```

bad filename ('name')

Macintosh file system error. You tried, either while running under System 6, to open a text file that has a name beginning with a period, or you have a problem with your file system or disk.

'bit' is not a legal align specifier

THINK Rez. You can't align a resource on a bit boundary. Only `nibble`, `byte`, `word`, and `long` are valid `align` specifiers.

bitstring is a numeric type

THINK Rez. Even though it sounds like a string it is actually a numeric type. A bitstring is used to define a sequence of bits. The type `bitstring[16]` is the same as an integer.

'break' not in loop or switch

The `break` statement must be inside a loop (`while`, `do-while`, `for`) or a `switch`. Example:

```
void function(void)
{
   short i;
   break;          // NO

   while(1)
      break;       // OK

   do { break; }   // OK
      while(1);

   switch(i){
   case 1:
      break;       // OK
   }
}
```

call of non-function

An expression in the function call position does not evaluate to a function or to a pointer to a function. Example:

```
void function(short (*f_ptr)())
                              // f_ptr is a
                              // pointer to a
                              // function.
{
  short i, j, k, result;
  result = i(j+k);            // NO: May be
                              //     missing
                              //     something
  result = (*f_ptr)(i,j);     // OK
  result = f_ptr(i, j);       // OK: Same as
}                             //     above.
```

can't add any more markers to this file

You have exceeded the limit of 65,535 markers in a file.

can't allocate stack temporary

The compiler tried to allocate a temporary variable beyond the first 32K of the local stack frame. THINK C uses temporary variables to store intermediate values in SANE floating-point operation and the return values of functions returning `struct` types. You should reduce the size of the local stack frame (by making large arrays

`static`, for example) or move the calculations into another function. Example:

```
void function(void)
{
   char a[32762];
   float x = 2.0;

   x = x * x + 3.0; // NO: Needs temp
                    //     beyond
}                   //     -32768(a6)
```

can't assign to array

You cannot assign, increment, or decrement arrays. Example:

```
void function(void)
{
  char a1[4];
  char a2[4];
  char a3[4];

  if (a1 == a2)   // OK: Compares array
                  //     addresses
      a1++;       // NO: Can't increment
                  //     arrays
  a1 = 0;         // NO: Can't assign to
                  //     an array
  a1 = a2;        // NO: Can't assign
}                 //     arrays
```

can't assign to function

You cannot assign, increment, or decrement functions. Example:

```
void function(void)
{
  short f1();
  short f2();
  short f3();

  if (f1 == f2)   // OK: compares function
                  //     addresses
    f1++;         // NO: can't increment
                  //     functions
  f1 = 0;         // NO: can't assign to a
                  //     function
  f1 = f2;        // NO: can't assign
}                 //     functions
```

can't calculate size of an expression that contains an array statement

THINK Rez. Statements that contain references to forward labels can't contain array statements.

can't calculate size of an expression that contains a switch statement

THINK Rez. Statements that contain references to forward labels can't contain switch statements.

can't call constructor *'symbol'* directly

You can invoke a class's constructor only by using new. You can't call it like a member function. Example,

```
class C {
public:
   C();
};

void function(void)
{
   register C *p, *q;

   p = new C;  // OK
   q = p->C(); // NO
}
```

can't close *'projectName'* while it is running

You are trying to close the project while it is running in a separate partition or under the debugger. You must quit the project first, then close it.

can't concatenate normal/wide strings

You can concatenate only wide strings with wide strings or normal strings with normal strings. Example:

```
char s1[] = "hello,"   "world".  // OK
char s2[] = "hello,"   L"world". // NO
char s3[] = L"hello,"  "world".  // NO
char s4[] = L"hello,"  L"world". // OK
```

can't convert 16-bit data references

.o Converter. The .o Converter can't convert data references that are 16 bits wide; recompile the source to the object file with the model Far option, or change the code so that the references are no longer necessary.

can't convert 32-bit A5-relative references

.o Converter. The .o Converter can only convert 16-bit A5-relative references, not 32-bit A5-relative references. If you need to support 32-bit references to code, recompile your object file source using the model Far option.

can't convert 32-bit code references unless "Far CODE" or "Far DATA" is enabled

.o Converter. The .o Converter can only convert 32-bit code references (ones that were created by using the model Far option) if you've enabled either the Far CODE or Far DATA options in the **Set Project Type** dialog box.

can't convert A5-relative data references

.o Converter. The .o Converter can't convert files that contain A5 relative references to data. Recompile the source to the object file so that either PC relative references are used, or with the model Far option.

can't convert computed references

.o Converter. The THINK Project Manager doesn't support certain computed references that contain:

- 8- or 32-bit computed references from data
- Any sized references to external data

To fix this problem, recompile the source to the object file so that these references are not generated.

can't convert "force active" modules

.o Converter. The Force active bit is a special bit that can be set from MPW C or MPW Pascal to prevent the MPW Linker from stripping out certain modules. The THINK Project Manager doesn't support this option, so files that use it cannot be converted. To fix the problem, recompile the source without the option set.

can't convert Object Pascal modules

.o Converter. The THINK Project Manager doesn't support the Object Pascal module bit that is set for certain Pascal Object classes. Either recompile the C++ source using Symantec C++ for Macintosh, or convert the code so that it doesn't use Pascal Object.

can't define classes in this project

Object Oriented Programming classes may be defined only in applications, or multi-segment DAs, DRVRs, or CODE resources. Check the Multi-Segment option in the **Set Project Type** dialog to use Object Oriented Programming classes.

Can't delete more than 65,535 characters at once.

You have too much text selected in the edit window. You won't be able to perform any edit operations until you select a smaller portion of text.

can't do that with multi-segment project

Your project is multi-segment, and what you're doing is allowed only with single-segment projects. You see this message in the following instances:

- Use the **Build Library** command on a multi-segment project
- Use a multi-segment project as a library
- Use the **Build Device Driver**, **Build Desk Accessory**, or **Build Code Resource** command on a multi-segment project when the Multi-segment option is not on

Can't find *tool*.

The THINK Project Manager cannot find the specified tool (ToolServer, SourceServer, Editor, or THINK Reference). You need to place the tool, or a valid alias to the tool, inside the `Tools` folder which is located in the same folder as THINK Project Manager. Also, the tool or its alias must have exactly the same name as appears in the error message (ToolServer, SourceServer, Editor, or THINK Reference).

can't find translator for *'filename'*

Your file has an extension that is not mapped to any translator, or the file has no extension and is not a library or project file. You can map extensions to translators in the Extensions page of the **THINK Project Manager options** dialog.

can't have data statement without matching type statement

THINK Rez. If you have a statement in a resource block, then there must be at least one type statement in the matching type block.

```
type 'test' {
   integer = 2;// constant statement
};
resource 'test' (128) {
   3;    //there are no statements in
         //type block that need a "3"!

};
```

can't load STRS in this project

You're trying to load a library into a project. You have set the Separate STRS option for the library on, and the separate STRS option for the project off. You need to rebuild the library with the Separate STRS off, or turn the option on in the project that's using the library.

can't modify 'const' location

You are trying to assign to a `const` variable, or to a location referred to by a pointer to a `const`. Example:

```
const short x = 0;// constant short
const short *p;   // pointer to
                  //   constant short
short * const q;  // constant pointer to
                  //   short

void function(void)
{
   x = 1;   // NO
   p = 0;   // OK
   *p = 0;  // NO
   q = 0;   // NO
   *q = 0;  // OK
}
```

can't open file *'filename'*

The THINK Project Manager cannot open the file in an `#include` or `$$resource` statement. Its name may be misspelled, or it may not exist, or it may be in the wrong tree. For more information, see Chapter 9, "Organizing Files and Folders."

can't open files of type *'typeID'*

THINK Project Manager received an Open Document AppleEvent, asking it to open a file that it doesn't know how to open.

can't open #included file
During compilation, THINK Project Manager encountered an `#include` statement specifying a file that couldn't be opened for one of these reasons: the file couldn't be found, the file is of the wrong type, or the file is corrupted.

can't open more than one project
THINK Project Manager can have only one project open at a time.

can't open text files if no project is open
You must open a project file before you can open any text files in THINK Project Manager.

can't print libraries
The THINK Project Manager received a Print Documents AppleEvent requesting it to print a library, which it cannot do.

can't run background-only project
You can't run your project under the THINK Project Manager if the Background only bit is set in its `'SIZE'` resource. You set that bit in the **SIZE Flags** menu in the **Set Project Type** dialog.

can't take address of method *'symbol'*
You can't take the address of a method unless it is `static`. Unlike C++, THINK C does not support pointers to non-`static` methods. Example:

```
class C {
public:
   void g(void);
   static void f(void);
} *cp;

void foo(void)
{
   void *p;

   p = &cp->g; // NO.
   p = &C::f;  // OK: f is static.
}
```

can't use 'far' code or data references in this project
Far CODE and Far DATA may be used only in Application projects, and only when the Far CODE and/or Far DATA check boxes in the **Set Project Type** dialog are checked.

cannot create file *'filename'*

THINK Project Manager tried to create a file with the same name as one that already exists.

'case' not in switch

You used the `case` keyword outside of a `switch` block.

char type only allows one character per string

THINK Rez. If the type statement is of type `char`, then the matching resource description statement needs to be in double quotes, but it must be zero or one byte long.

```
""    //valid char,
      //  a zero byte will be output
"a"   //valid char
"ab"  //invalid char
```

character literal must be 1, 2, or 4 bytes

A character literal must contain exactly 1, 2, or 4 characters. Example:

```
char c1 = 'A';          // OK
char c2 = 'AB';         // OK
char c3 = 'ABC';        // NO
char c3 = 'ABC ';       // OK
char c3 = '\n\n\n\n';   // OK
```

class declarations must appear outside any function

You must declare a class globally. You cannot declare it within the body of a function.

class must have a name

Unlike a `struct`, `union`, or `enum`, a `class` declaration must have a tag name. Example:

```
typedef class {      // NO.
   void foo(void);
} C;

class C {           // OK: Compiler inserts
   void foo(void);  //     implicit
};                  // 'typedef class C C'.
```

code overflow

You have more than 32K of code or data in one file. Break your file into smaller files.

code segment too large

This error occurs when the file being compiled produces over 32K of code and will not link correctly. Try splitting up the code into smaller sections or turn on the Always resegment on overflow option on the Preferences page of the **THINK Project Manager options** dialog.

constant required

THINK C expected a constant but didn't find one. Example:

```
#define aConst  12
void function(void )
{
  short h;
  short x[2];     // OK: 2 is a constant
  short y[h];     // NO: h isn't a constant
  enum {
     color = h,   // NO: enum values must
                  //     be constant
     size = aConst// OK
  } attributes;

  struct {
      unsigned illegal_bitfield : h;  // NO
      unsigned legal_bitfield : size; // OK
  } bits;

  short z[size];  // OK: Array dimension is
 }                //     a constant
```

contains a function call

Debugger. The expression in the Data window contains a function call. In Figure C-1, entering `rand()` displays an error message, but entering `rand` displays the address of the function.

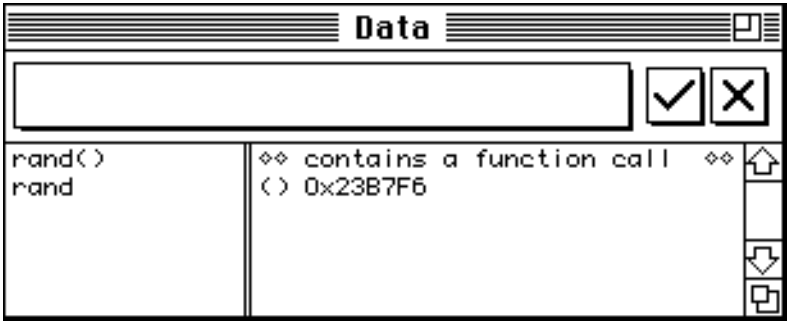

Figure C-1 The "contains a function call" error message

To evaluate expressions in the Data window, turn on the Allow dataview function calls option on the Debugging page of the **THINK Project Manager options** dialog.

'continue' not in loop

The `continue` statement must be inside a loop (`while`, `do while`, `for`). Example:

```
void function(void)
{
  short i;
  continue;          // NO

  while(1)
      continue;      // OK

  do { continue; }   // OK
      while(1);

  switch(i){
  case 1:
      continue;      // OK
  }
}
```

could not evaluate constant expression

An expression that's in a position where THINK C requires a constant (such as an enumeration constant or an array size) is too large for the compiler to evaluate. Example:

```
enum { foo = 1 << 33 }; // NO.

void function(void)
{
   short x;
   x = 1 << 33;          // OK: Yields 0 at
}                        //     runtime.
```

cstring too long (will be truncated)

THINK Rez, Warning. If no length indicator is given, a `cstring` appends a null byte to the string. The maximum size of a `cstring` is 2,147,483,647, but THINK Rez will run out of memory long before then.

data segment too large

This error occurs when a program is compiled that declares more than 32K of global data. Either allocate the data dynamically, or split

up the declarations between multiple files and use the Far data option.

data statement type does not match type statement declaration
THINK Rez. Each data statement must have a type statement of the same base type.

base type	instance types
`string`	`string, cstring, pstring, wstring,char`
`bitstring`	`boolean, byte, integer, longint,bitstring`
`rect`	`rect`
`point`	`point`

It is an error if the instance type (the resource statement) does not match the base type in the type statement.

declaration of *'symbol'* does not match #pragma parameter
There is a `#pragma parameter` directive for a function named *symbol*, and the number or types of arguments in the prototype do not match those in the pragma. Example:

```
#pragma parameter __A0 myfunc(__D0)

void myfunc(long);          // NO
short myfunc(long);         // NO: Can't return
                            //     short in A0
void *myfunc(long, ...);    // NO: Can't use
                            //     '...'
long myfunc(Rect r);        // NO: Can't
                            //     pass
                            //     struct in D0
pascal long myfunc(Rect r);// OK: Passes
                            //     address
                            //     of Rect
```

declarator too complex
The declaration statement is too complex. You're unlikely to get this error unless you're compiling a computer-generated program.

'default' not in switch
You used the `default` keyword outside of a `switch` block.

different configuration for precompiled header
You're using a precompiled header that you precompiled with different options than you're using now. Precompile the header

again with the options you're using now, or check your current options to make sure they're correct.

directory *'dirName'* not found

Standard Macintosh system error. You will usually get this error whenever THINK Project Manager is searching for a file or folder (opening a project, saving a file, compiling, searching with the **Find** command, and so on). Make sure that you use the correct directory name.

directory is full

THINK Project Manager tried to create a file in a folder that already contains as many files as it can hold.

disk is full

There are two possible causes for this error:

- The THINK Project Manager tried to create or save a file on a disk that doesn't have enough free space to accommodate the file.
- The .o Converter ran out of disk space when trying to create a vocabulary file. Either turn off the Create .v file option, or clear off enough space on the disk that contains the `.o` file for the vocabulary file.

displacement too large for this addressing mode

Assembly. The displacement portion of an addressing mode is out of range for that mode. Usually, you areusing an indexed addressing mode with a displacement less than -128 or greater than 127. Example:

```
asm {
   clr.w 200(a0,d0.w)    ; NO.
   clr.w 0xFE(a0,d0.w)   ; NO: Hex numbers
                         ;     are unsigned.
   clr.w -0x02(a0,d0.w)  ; OK.
   clr.w 100000(a2)      ; NO: Displacement
                         ;     must fit in
                         ;     16 bits.
   clr.w (100000,a2)     ; OK, If MC68020
}                        ;     is on.
```

duplicate 'case'

Within a `switch` block, there are two or more `case` expressions with the same value.

duplicate 'default'
You have more than one `default` in a `switch` block.

duplicate register in argument list
You used the same register more than once in the argument list of a `#pragma parameter` directive. Example:

```
#pragma parameter __A0 f(__D0, __D0);  //NO
#pragma parameter __A0 g(__A0);        //OK
```

duplicate resource 'type',id
You have multiple copies of this resource coming from various `.r` and `.rsrc` files in your project. (There could also be one in your `project_name.rsrc` file.)

'enum *symbol*' already defined
There is already an `enum` with the tag *symbol*. Example:

```
enum Days { Monday, Tuesday, Wednesday,
            Thursday, Friday };
enum Days { Mothers, Fathers, Secretaries,
            Memorial, Labor };
```

'enum *symbol*' already in use
The `enum` tag *symbol* is already used as a `struct` or `union` tag. Example:

```
typedef struct X { short x; float y; };
typedef enum X { a, b, c, d };
```

enumeration constant too large
The value of an enumeration constant is too large for a 4-byte `int`. This error happens only if the Enums are always ints option is on

and 4-byte ints option is off. Values that cannot be represented in 32 bits bring up the error “could not evaluate constant expression”.

#error directive

The preprocessor came across an `#error` directive. You can use the `#error` directive to report on any potential problems that the preprocessor can detect. Example:

```
#if __option(trigraphs)
   #error Trigraphs confuse File Manager.
#endif

#define FILE_TYPE    '????'
#define FILE_CREATOR '????'
```

error opening resource file for $$resource or $$read statement

THINK Rez. The `$$resource` and `$$read` statements require THINK Rez to be able to open the resource fork of a file. Make sure that the file exists, that it is not opened with write access by another running application, and that it is in the system or project tree.

error reading resource map

The Resource Copier encountered a Resource Manager error when trying to open the source resource file. Try using ResEdit’s Verify option to repair the resource map in this file.

escape sequence doesn’t fit in a byte

An escape sequence is a series of two or more characters that form a character. Octal, hexadecimal, decimal and binary escape sequences can be used to specify characters that do not have predefined escape sequences. However, all escape sequences must fit into a single byte. Escapes must be of the form:

base	number form	digits	example
2	\0Bbbbbbbbb	8	\0B10000001
8	\000	3	\101
10	\0Dddd	3	\0D065
16	\0Xhh	2	\0X41
16	\$hh	2	\$41

Here are some examples:

```
\077           /* 3 octal digits*/
\0xFF          /* ‘0x’ plus 2 hex digits*/
\$F1\$F2\$F3   /* ‘$’ plus 2 hex digits*/
\0d099         /* ‘0d’ plus 3 decimal digits*/
```

Remember that an octal escape code consists of exactly three digits. For instance, to place an octal escape code with a value of 7 in the middle of an alphabetic string, write `AB\007CD`, not `AB\7CD`.

Also, remember that THINK Rez is not C or C++. The string "`\0`" is not valid in THINK Rez, because the `\0` character is not a valid escape sequence.

expected point type data statement
THINK Rez. As THINK Rez parses your resources, it knows when to expect certain kinds of statements. You will get this error if THINK Rez expects to parse a point statement, and sees something other than a point statement.

expected switch statement
THINK Rez. As THINK Rez parses your resources, it knows when to expect certain kinds of statements. Make sure you picked a valid case from the switch statement in the matching type block.

expression too complex
Usually, there are not enough scratch registers to evaluate a complex addressing expression. Try splitting the expression into subexpressions. You can also disable automatic register allocation for the function so more registers are available. To disable automatic register allocation, use this line:

```
#pragma options(!assign_registers)
```

file *'filename'* is already open
You tried to use the **Save** or **Build** command to replace an existing file that is currently open. Or, THINK Project Manager tried to delete a file that is currently open.

file *'filename'* not found
Standard Macintosh system error. Can occur whenever THINK Project Manager is searching for a file (for instance, when compiling, or when searching with the **Find** command).

first argument to function does not match prototype
first argument to function *'symbol'* does not match prototype
The first argument to a function does not match the type in the prototype. The prototype may be explicit or inferred.

floating constant too large

A floating-point constant was too large (or too small) to be represented. For various limits of floating-point types, see the online *Standard Libraries Reference*. Example:

```
long double x = 1.2e100000;
      // NO: Maximum exponent is 4931.
```

grep error: *xxx*

Something is wrong with the grep expression you entered in the **Find** dialog. The error string *xxx* may be any one of the following:

- Pattern too long
- No "`]`" found to match "`[`"
- Too many occurrences of "`(`"
- Too many occurrences of "`)`"
- No "`)`" found to match "`(`"
- Too few subexpressions to match "<*digit*>"

has side-effects

Debugger. You entered an expression that has a potential side effect: `=`, `++`, or `--`. In Figure C-2, the debugger won't evaluate `i++` and `i+=1` because they have side effects.

Figure C-2 The "has side-effects" error message

hex strings must be an even number of digits

THINK Rez. All hex strings must be an even number of digits. THINK Rez will not pad the string for you.

I/O error on file *'filename'*

A Macintosh error occurred while the file was being read or written, that is, saved. The read or write operation may not have been completed successfully.

illegal #elif

illegal #else

Every `#else` or `#elif` must have a matching `#if`, `#ifdef`, or `#ifndef`. Example:

```
#ifdef name
#else
#else            // NO: No matching #ifdef
#endif
#elif condition  // NO: No matching #if
```

illegal #endif

Every `#endif` must have a matching `#if`, `#ifdef`, or `#ifndef`.

illegal application of '&'

The address-of operator was used on an object (such as a register variable or a bit-field) that doesn't have an address. It is not an error to take the address of an inline function. The compiler generates a reference to a function with the same name, and brings up the error "link failed". Example:

```
pascal void Debugger(void) = 0xA9FF;
void function(void)
{
  register short i;
  short *p = &i;         // NO: i is a
                         //     register
  void *q = &Debugger;   // OK: But
}                        //     probably
                         //     not what
                         //     you meant
```

illegal application of 'pascal'

Only functions or function types can be declared `pascal`. Example:

```
pascal short f(void);    // OK.
pascal short i;          // NO.
pascal short *p(void);   // OK: pascal
                         //     function
                         //     returning
                         //     ptr to
                         //     short.
pascal short (*q)(void);// OK: pointer to
                         //     pascal
                         //     function
                         //     returning
                         //     short.
```

illegal application of 'sizeof'

You cannot use `sizeof` with a function or a function type. Example:

```
short f(void), (*g)(void);

short a = sizeof(f); // NO
short b = sizeof(g); // OK: b = sizeof
                     //     pointer
short c = sizeof(*g);// NO
```

illegal array bounds

The bounds of an array can't be negative or zero. Examples:

```
short a[-7];   // NO: bounds can't be
               //     negative
short b[0];    // NO: bounds can't be zero
```

illegal array element type

You can't declare an array of functions, or of `void` objects. However, you can declare an array of pointers to functions or of pointers to `void`. (`void *x[]` is a C idiom for an array of generic pointers.) Example:

```
short array_of_functions[]();       // NO
void abyss[SIZE];                   // NO
void *generic_ptr_array[SIZE];      // OK
short (*array_of_ptrs_to_funcs[SIZE])();
                                    // OK
```

illegal bitstring size

THINK Rez. The only numeric type specification that allows a size is `bitstring`. The `byte`, `integer`, and `longint` types cannot have sizes.There is no implicit array type as in C. If you want an array of bytes, you have to explicitly declare them:

```
type 'test' {
   array [12] {
      byte; //12 bytes
   };
};
```

illegal cast

You cannot cast structs or unions to other types. You can cast numerical values or pointers to other numerical values or pointers. Example:

```
typedef struct {short v, h;} Point;
void function(void)
{
  Point p;
  long l;
  p = (Point) l;         // NO
  p = *(Point *) &l;     // OK
}
```

illegal character *'character'*

A character that's not part of the C character set (such as @ or $) appears in your source file outside of a comment or string. If the character isn't a printable ASCII character, the error message shows it's hexadecimal value. Example:

```
asm {
   dc.w  $A9FF       ; NO: Use 0xA9FF.
}
```

illegal floating-point operation

You cannot use some of the C operators on floating-point values. Example:

```
void function(void)
{
  double d1;
  double d2;
  double d3;
  d3 = d1 % d2;   // NO: Can't do floating
                  //     modulo
  d3 = d1 << 3;   // NO: Can't do floating
                  //     shifts
  d3 = 3 << d1;   // NO: Can't do floating
}                 //     shifts.
```

illegal function return type

A function cannot return an array or a function. However, a function can return a pointer to an array or a return pointer to a function. Example:

```
typedef char Str255[256];
typedef short (* function_ptr)();

Str255 function_returning_array();  // NO
Str255 *function_returning_ptr_to_array();
                                    // OK
short function_returning_function()();
                                    // NO
function_ptr  f_returning_ptr_to_f();
                                    // OK
```

illegal hex string character

THINK Rez. Escape sequences are not allowed inside hex strings.

illegal initialization

If the project is a code resource, desk accessory, or device driver, you cannot initialize an address in global or static memory. Example:

```
static short i;
static short *p = &i;   // NO: Not in
                        //application
```

illegal near reference: *'symbol'*

See the error message “link failed” later in this appendix.

illegal operation on array
illegal operation on function
illegal operation on struct
illegal operation on union

You cannot use many C operators on structs or unions. Example:

```
short function(void)
{
  struct {
    short x;
  } s1, s2, s3;
  short i;

  if (s1 == s2)   // NO: Can't compare
                  //     structs
    s1++;         // NO: Can't increment
                  //     structs
  s1 = 0;         // NO: Can't assign
                  //     short to a struct
  s1 = s2;        // OK: struct assignment
                  //     is allowed
  s3 = s1 + s2;   // NO: Can't add structs
  i = s1 - s2;    // NO: Can't add structs
  i = (long)s1;   // NO: Can't cast a
                  //     struct
  return(s1)      // NO: return does an
}                 //     implicit cast
```

illegal pointer arithmetic

You cannot perform some arithmetic operations on pointers. You cannot assign integers to pointers except for the constant zero. In addition, you cannot compare pointers to integers, again, with the

exception of the constant zero. In all cases, you can use a cast to force the operation when necessary. Example:

```
void function(void)
{
  short *short_ptr;
  short i;
  char *p1, *p2;
  long result;

  short_ptr = 0x220;     // NO
  short_ptr = 0;         // OK
  short_ptr = (short *) 0x220;
                         // OK: Address of
                         //     MemErr
  result = p1 + p2;      // NO: Can't add
                         //     pointer
  result = p1 / p2;      // NO: Can't divide
}                        //     pointers
```

illegal preprocessing directive

These are the valid preprocessing directives, any other identifier preceded by a `'#'` is an error:

```
#if          #ifdef          #ifndef
#elif        #else           #endif
#include     #define         #undef
#line        #error          #pragma
```

illegal rect statement

THINK Rez. If the `type` statement called for a rectangle, then your array must contain exactly four elements.

```
{ 0, 0, 100, 100 };     /* valid rect */
{ 0, 0 };               /* illegal rect */
```

illegal redefinition of macro ‘symbol’
illegal resource attribute

THINK Rez. THINK Rez can’t set the changed bit of a resource.

illegal size for bitfield

Bit-field sizes must be less than or equal to the number of bits in the type specifier. Labeled bit-field sizes must be greater than zero. Unlabeled bit-field sizes of 0 force a word alignment. Example:

```
struct bitfields{
  char c : 9;     // NO: chars are
                  //     8-bits long
  short i : 17;   // NO: shorts are
                  //     16-bits long
  short zero : 0; // NO: Bitfield size
                  //     must be over 0.
  short good : 11;// OK
  short : 0;      // OK: Forces word
                  //     alignment
  long l : 33;    // NO: longs are
                  //     32-bits long
  short z : -4;   // NO: Bitfield size
};                //     must be over 0.
```

illegal type for bitfield

These are the only types allowed for a bit-field:

```
char                  short int
int                   long int
unsigned char         unsigned short int
unsigned int          unsigned long int
```

You cannot use any other type for a bit-field. Example:

```
struct bitfields{
  char c : 5;           // OK
  unsigned int i : 5;   // OK
  long l : 17;          // OK
  double d : 10;        // NO
  void *p : 8;          // NO
};
```

illegal type for field

A field may not be a function, unless it is a field of a class. Example:

```
struct S {
   short f();  // NO: S is not a class
};

struct T : indirect {
   short f();  // OK: T is a class
};
```

illegal type id range

THINK Rez. If a range is provided by a type declaration, then all resources of that type must be within that range. It is legal to have several type declarations of various ranges, but all resources must be within one of those ranges.

```
type ‘test’ (0:128){
   pstring;
   rect;
};
type ‘test’ (129:131){
   pstring;
   point;
};

resource ‘test’(128) {//legal - falls within
                      // range of first
                      // 'test' type
   "hello world",
   { 0, 0, 100, 100 };
};

resource ‘test’ (1000) {//no type exists for
                       // 'test' id = 1000
   "hello world",
   { 0, 0 };
};
```

illegal use of class type

You must declare an object as a pointer. Also, you can’t use `sizeof` with a class. Example:

```
void function(void)
{
   ClassName   badObject;           // NO
   ClassName   *goodObject;         // OK
   char        a[sizeof(ClassName)];// NO
}
```

illegal use of void

You cannot use most of the C operators on `void` values. You cannot declare variables of type `void`. Example:

```
void f(void)
{
  short i, j;
  i = (void)j + 5; // NO: Can't add a void.
  i = f() - 3;     // NO: f() returns void.
}
```

immediate value out of range

Assembly. An immediate operand is outside the range of permissible values for the instruction. Example:

```
asm {
   addq.l#10, d0     ; NO: ADDQ/SUBQ range
                     ;     is 1..8
   lsr.l #10, d0     ; NO: LSR/LSL range
                     ;     is 1..8
   moveq #128, d0    ; NO: MOVEQ range
                     ;     is -128..127
}
```

#include files nested too deeply

You tried to compile a file that had `#include` directives nested more than 50 levels deep. Usually, one of your header files includes another header file which (eventually) includes the original file again. You can use the `#pragma once` directive to inhibit recursive processing of include files.

incomplete macro call

THINK C reached the end of file before a macro call completed. Example:

```
#define macro(arg1, arg2) (arg1 > arg2)
macro(x,          // NO
```

incorrect base register

Assembly. You're probably referencing a global through a register other than `A5` (for applications) or `A4` (for desk accessories, device drivers, or code resources). You don't need to provide a base register to use a global variable in assembly language. Example:

```
short theGlobe;      // a global

void OffBase(void)
{
   asm{
      move  theGlobe(a2), d0  // NO
      move  theGlobe(a5), d0  // OK: But
                              //     not
                              //     needed
      move  theGlobe, d0      // OK: Much
   }                          //     better

}
```

instruction requires 68020
instruction requires 68881

Assembly. You're using an MC68020 or MC68881 instruction but MC68020 or MC68881 inline assembly is not enabled. Example:

```
asm 68000 {
   extb.l   d0    ; NO: EXTB.L requires
                  ;     68020
   ext.w    d0    ; OK: Use EXT.W/EXT.L on
   ext.l    d0    ;     68000
}
```

integer constant expression expected

You must use an integer constant expression in `case` statements, in array size declarations, and in the `#if`, `#elif`, `#exit`, and `#line` preprocessor commands.

integer constant too large

The valid range for values of `byte`, `integer`, and `longint` are as follows:

type	maximum	minimum
byte	255	-128
integer	65,535	-32,768
longint	4,294,967,295	-2,147,483,648

invalid 68881 unary operator *'symbol'*

You must declare an MC68881 unary operator to take an extended precision argument and return an extended precision result. See `math.h` for examples of MC68881 unary operators. Example:

```
// The "8-byte double" option is on.

float acos(float) : 0x1C;           // NO
double acos(double) : 0x1C;         // NO
long double acos(long double) : 0x1C;
                                    // OK
```

invalid access specifier *'symbol'*

symbol is not `public`, `private`, or `protected`.

invalid addressing mode

Assembly. You specified an operand to an assembly language instruction which is not a valid MC68000 or MC68020 addressing mode. Example:

```
asm {
   clr.w    (d0)+       ; NO
   move.w   (pc),d0     ; NO
   clr.w    -2(d0,a0)   ; NO
   clr.w    -2(a0,d0)   ; OK
}
```

invalid constructor

You tried to declare a return type for a constructor, or tried to declare a member function with the same name as its class. Example:

```
class C {
public:
   void C();   // NO: Constructor can't
               //     have return type
   C();        // OK
};
```

invalid declaration

A syntax error is in a declaration. Example:

```
short i j;     // NO: Missing ','
```

invalid declaration of 'operator delete'

The prototype for `operator delete` must be like this:

```
void operator delete(void *);
```

invalid declaration of 'operator new'

The prototype for `operator new` must be like this:

```
void *operator new(size_t);
```

If you are not using `stddef.h`, you can use `long` or `unsigned long` in place of `size_t`.

invalid destructor

You tried to declare arguments for a destructor. Example:

```
class C {
public:
   ~C(long);   // NO
   ~C(void);   // OK: But void is unneeded
   ~C();       // OK: Much better
};
```

invalid escape character

THINK Rez. THINK Rez does not allow just any character to be escaped. It expects them to be formatted as octal, hex, decimal or binary escapes. An escape sequence is a series of two or more characters that form a character. Octal, hexadecimal, decimal and binary escape sequences can be used to specify characters that do not have predefined escape sequences. However, all escape sequences must fit into a single byte. Escapes must be of the form:

Base	Number form	Digits	Example
2	\0Bbbbbbbbb	8	\0B10000001
8	\000	3	\101
10	\0Dddd	3	\0D065
16	\0Xhh	2	\0X41
16	\$hh	2	\$41

Here are some examples:

```
\077           /* 3 octal digits*/
\0xFF          /* ‘0x’ plus 2 hex digits*/
\$F1\$F2\$F3   /* ‘$’ plus 2 hex digits*/
\0d099         /* ‘0d’ plus 3 decimal digits*/
```

Remember that an octal escape code consists of exactly three digits. For instance, to place an octal escape code with a value of 7 in the middle of an alphabetic string, write `AB\007CD`, not `AB\7CD`.

THINK Rez is not C or C++; the string `"\0"` is not valid in THINK Rez, because the `\0` character is not a valid escape sequence.

invalid floating-point literal

Assembly. A quoted floating-point literal is not a legal SANE floating-point number. Example:

```
asm {
   dc.x  "nan0"      ; NO
   dc.x  "nan(0)"    ; OK
}
```

invalid function prototype

You used a bad function prototype. If you get this error, it is usually because the Language Extensions option is off, and the function prototype contains ". . ." which is one of the THINK C language extesions.

invalid number

You specified an illegal integer or floating-point number. Example:

```
short x = 0xA9FG;        // NO
float f = 1e*37;         // NO: But +37
                         //     or -37 is
                         //     OK.
integer = 123a;          //NO
longint     zero = 0, one = 1two = 2;
                         // invalid number
                         // 1two
```

invalid object file

.o Converter. This error occurs when the object file that the .o Converter is attempting to convert contains an incorrect MPW Object File record. To determine where the bad record occurs in the file, you can use the MPW tool "RezDet".

invalid parameter to #print directive

THINK Rez. The `#printf` directive can be used to print strings and integral numbers. It can't be used to print array values or label indexes.

invalid redeclaration of *'symbol'*

You declared an identifier twice, and the second declaration is incompatible with the first. Example:

```
extern short x;
long x;              // NO: x is redeclared
                     //     differently
extern short y;
short y;             // OK: y is redeclared
                     //     as the same
                     //     type
short z;
short z;             // NO: Can only have
                     //     one defining
                     //     instance
typedef short IsARose;
IsARose IsARose;     // NO
```

invalid redefinition of macro *'symbol'*

You tried to `#define` an already defined macro with a different definition. You can always redefine a macro to the same thing. If you want to actually change the definition of the macro, you must `#undef` it first. Example:

```
#define NULL 0L
#define NULL 0L             // OK
#define NULL ((void *) 0)   // NO
#undef NULL
#define NULL ((void *) 0)   // OK
```

invalid register *'symbol'*

The name *symbol* is not one of the permitted register names for a `#pragma parameter` directive. You can use only these registers:

```
A0          A1
D0          D1              D2
```

invalid register list

Assembly. The register list contains duplicates, or the list is in the wrong order. Example:

```
asm {
   movem a0/a0,-(sp) ;NO
   movem d7-d1,-(sp) ;NO
   movem d1-d7,-(sp) ;OK
}
```

invalid scale factor

Assembly. In an MC68020 indexed addressing mode, you can apply a scale of only 1, 2, 4, or 8 to the index register. Example:

```
asm 68020 {
   clr.w 20(a0,d0.w*0)        ; NO
   clr.w 20(a0,d0.w*3)        ; NO
   clr.w 20(a0,d0.w*(3+1))    ; OK
   clr.w ([20,a0],d0.w*8,10)  ; OK
}
```

invalid size

Assembly. In an assembly language instruction, you used a bad size specifier or the wrong size specifier for what you're trying to do. Example:

```
asm {
   move.b   a0,a1    // NO: Can't move
                     //     bytes to
                     //     address
                     //     registers.
   move.q   a0,a1    // NO: .q is not a
}                    //     legal size.
```

invalid storage class

You applied incompatible attributes to a data item, implicitly or explicitly. Example:

```
register short aGlobal; // NO: register
                       //     global
extern auto short y;   // NO: contradic-
                       //     tory storage
                       //     class
```

invalid type specification

You used an illegal or ambiguous combination of type keywords when declaring a data item. Example:

```
unsigned double d;         // NO
long struct {short i;} s;  // NO
```

invalid use of class specifier

You tried to declare a member of a class outside of a `class` declaration. Example:

```
class C {
public:
   static short i;
   short j;
   void foo(void);
};

short C::j = 0;          // NO
short C::i = 0;          // OK: Initializes
                         //     class
                         //     variable.
void C::foo(void);       // NO
void C::foo(void) {…}    // OK
```

invalid use of 'operator delete'

You tried to declare `operator delete` outside of a `class` declaration. To define `operator delete` for a class, you must prefix it with the class name and `::`. Example:

```
class C {
   void operator delete(void *);
};

void *operator delete(void *p) {…}  // NO
void *C::operator delete(void *p) {…}
                                    // OK
```

invalid use of 'operator new'

You tried to declare `operator new` outside of a `class` declaration. To define `operator new` for a class, you must prefix it with the class name and `::`. Example:

```
class C {
   void *operator new(size_t);
};

void *operator new(size_t n) {…}    // NO
void *C::operator new(size_t n) {…} // OK
```

jump table too big

See the error message "link failed" later in this appendix.

label *'symbol'* is already defined

The label *symbol* has been defined twice within the function.

last argument to function does not match prototype
last argument to function *'symbol'* does not match prototype

The last argument to a function does not match the type of the argument in the prototype. The prototype may be explicit (it must be explicit if the Require prototypes option is set) or inferred (if the Infer prototypes option is set).

link failed

The linker couldn't link your program. Usually, the linker fails due to an undefined symbol or to a multiply defined symbol (a symbol defined more than once in a project). To find out which files contain the offending symbols, use the **Check Link** command in the **Project** menu. The Link Errors window displays the names of the files in parentheses.

- **code segment too big** One or more of your code segments has exceeded the 32K limit. See "Segmenting Projects" in Chapter 8.
- **data segment too big** You declared more than 32K of global and static data in your project. Use memory allocation to create large data structures or turn on the Far DATA option in **Set Project Type** dialog. See "Using more global data with Far DATA" in Chapter 13.
- **illegal near reference: '*symbol*'** A 16-bit reference was made to a variable or function which was not in the first 32K of the data area or jump table, probably from a library. Recompile the library using the Far Code or Far Data options. See "Mixing Near CODE and DATA with Far CODE and DATA" in Chapter 13.
- **jump table too big** If the Far CODE option is off, your jump table is over 32K. If the Far CODE option is on, your jump table is over 256K. See "Using a larger jump table with Far CODE" in Chapter 13.
- **multiply defined: '*symbol*'** The *symbol* was defined more than once.
- **resource too big** For drivers, DRVR + DATA resource is > 32K. For multi-segment drivers, DATA + JUMP > 32K. For code resources, CODE + DATA + JUMP > 32K. For multi-segment code resources, CODE + DATA + JUMP >

32K for segment containing `main()`, CODE > 32K for other segments.

- **undefined: '*symbol*'** You referred to *symbol* without defining it.
- **undefined: *class*::** You didn't define the first method you declared in *class*.
- **undefined: ?*class*::*method*** You declared a method in your class declaration but didn't define it. You might get this message if you defined *method*, but you didn't define the first method you declared in *class*.

lvalue required

An lvalue is an expression that refers to an object in memory that you can store to and examine. Example:

```
short short_f(void);
short *pshort_f(void);
void function(void)
{
  short i;
  short *p = &i;

  // Operand of ++ must be an lvalue
  7++;            // NO
  short_f()++;    // NO
  pshort_f()++;   // NO

  // Left operand of an assignment
  // must be an lvalue.
  short_f() = i;  // NO
  pshort_f() = i; // NO
  *pshort_f() = i;// OK: Produces an lvalue
  (*p)++;         // OK
  (*pshort_f())++;// OK
}
```

macro argument *'symbol'* appears more than once

The formal parameters in a macro definition must appear only once. Example:

```
#define macro(again,again) (again+7)  // NO
```

memory critical - proceed at your own risk

Now would be a *really* good time to save your files and quit. Before you work on your project again, you may want to increase the size of THINK C's partition with the Finder's **Get Info** command.

missing ';'

THINK C expected a semicolon, but couldn't find one. Sometimes THINK C cannot determine that a semicolon is missing, and displays the error "syntax error".

missing 'key' in case selector

THINK Rez. Every case in a switch statement must have a key statement. It is this key statement that allows THINK Rez to select the correct case. Which case applies is based on the key value, for example,

```
type 'DITL' {
   ...type specifications...
   switch {
      case Button:
         boolean        enabled, disabled;
         key bitstring[7] = 4;/* key value */
         pstring;
      case CheckBox:
         boolean        enabled, disabled;
                  /* missing key statement */
         pstring;
      case ...
   };
};
```

missing #endif

Every `#if`, `#ifdef`, and `#ifndef` must have a matching `#endif`.

missing argument name

The argument list in a function definition did not name all the arguments. Example:

```
void foo(short i, short j, float);
                                  // Prototype

void foo(short i, short j, float) { … }
                                  // NO

void foo(short i, short j, float f) { … }
                                  // OK
```

missing or undefined label or enumerated value

THINK Rez. THINK Rez expected an identifier, but got something else like a ',' or '}'.

multiply defined: *'symbol'*

See the error message "link failed" described earlier in this appendix.

No classes are declared

This message appears in the Class Browser window when there are no classes declared in the current project.

no files in project

Your project must contain at least one file to accomplish what you're trying to do.

no input file specified

In this command line, you must include an input file.

no members defined

You defined a `struct` or `union` without members. Example:

```
struct memberless_struct{ };  // NO
union memberless_union { };   // NO
```

no methods defined

A class definition must have at least one method.

```
struct BadName : indirect {
   short instanceVar;   // NO: No methods
};                      //     declared
```

no resources read by include statement

THINK Rez, Warning. If an `include` statement fails for any other reason other than running out of memory, you will get this error message. Make sure that the file exists, that it is not opened with write permission by any other application, and that you have specified a valid pathname.

no resources read by read statement

THINK Rez, Warning. If a `read` statement fails for any other reason other than running out of memory, you will get this error message. Make sure that the file exists, that it is not opened with write permission by any other application, and that you have specified a valid pathname.

no such member *'symbol'*

In a `.` or `->` expression, the right operand is not the name of a member of the class or struct for the left operand. Example:

```
void function(void)
{
  struct s1_struct{
      short member_1;
   }s1, *p1;
  struct s2_struct{
      short member_2;
   }s2, *p2;
  s1.non_member;      // NO: non_member is
                      //     not in
                      //    s1_struct
  p1 -> non_member;   // NO: non_member is
                      //     not in
                      //    s1_struct
  s1.member_2;        // NO: member_2 is
                      //     not in
                      //     s1_struct
}
```

no such option *'symbol'*

You used the identifier *symbol* as an argument to `#pragma options` or `__option`, but it is not one of the valid options. Example:

```
#pragma options(!optimize)          // NO
#pragma options(!global_optimizer)  // OK
```

not enough bits to use $$byte, $$word or $$long

THINK Rez. If the resource is smaller than 8 bits, you can't use `$$byte`. Similarly, you need 16 bits and 32 bits to use `$$word` and `$$long`.

not enough memory to open header file '*filename*'

The THINK Project Manager can't load the precompiled header file. Try increasing the THINK Project Manager's memory (heap) partition to enable it to load the file.

nothing was declared!

You tried to compile an empty source file. If the Language Extensions option is off, source files must contain at least one declaration.

object file too large

This error is generated when a .o file produces more than 32K of CODE or DATA. Due to limitations in the Project Manager, imported .o files can only contribute up to 32K of CODE and 32K of DATA.

object pointer required

The operand of a `delete` or `__classof` operator has a type which is not a pointer to an object. Example:

```
void function(void)
{
   void *p = new A;  // p is a generic
                     //   pointer
   A *q = new A;     // q is an object
                     //   pointer

   delete p;         // NO
   delete q;         // OK
   delete (A*)p;     // OK
}
```

object pointer required for *'symbol'*

You are trying to access a non-static member of a class, and you did not specify an object pointer. Example:

```
class C {
public:
   static short i;
   short j;
   static void f(void);
   void g(void);
};

void foo(C *cp)
{
   C::i = 0;   // OK: 'i' is static.
   C::j = 0;   // NO
   cp->j = 0;  // OK
}

void C::g(void)
{
   C::j = 0;   // OK: 'this' is implied.
}
```

only one precompiled header allowed per file

This error occurs if a source file includes more than one precompiled header file. The THINK Project Manager only allows one precompiled header file per project. This error also occurs if a

source file includes a precompiled header after including a text header file. The precompiled header must occur first.

option *'symbol'* can not be adjusted

Some of the compiler options can not be set or cleared with the `#pragma options` directive during compilation. Usually these are project-wide options, such as `int_4` or `a4_globals`. You can change these options for the entire project from the **Options** and **Set Project Type** dialogs.

option *'symbol'* can not be adjusted within a function

With the `#pragma options` directive, you cannot set some compiler options within the body of a function. Move the pragma before the body of the function. Example:

```
void function(void)
{
#pragma options(check_ptrs)   // NO
   …
}

#pragma options(check_ptrs)   // OK
void function(void)
{
   …
}
```

out of memory

This error has several possible causes:

- The THINK Project Manager ran out of memory. Close open windows to reclaim more memory. If you get this message when you try to run the debugger, try making your application's partition size smaller. See "Memory Considerations" in Chapter 12.
- The Resource Copier ran out of memory when trying to copy a resource. This can happen with really large resources. To fix this error, increase the partition size of the THINK Project Manager.
- The .o Converter ran out of memory when trying to copy a resource. This can happen with very large object files. To fix this error, increase the partition size of the THINK Project Manager.
- A translator ran out of memory. You need at least enough memory to load the source file (including all `#include` files) and the compiled resources themselves.

pointer required

You're using an operator that requires a pointer with an expression that's not a pointer. Example:

```
void function(void)
{
  short x, result;
  result = *x;          // NO: x isn't a
                        //     pointer
  result = x->a_member; // NO: x isn't a
}                       //     pointer
```

pointer types do not match

You tried to assign, compare, or subtract incompatible pointers. In THINK C, pointers are more strictly typed than some other C compilers. In assignments and comparisons, the two pointer types must match or be of type `void *`. When subtracting two pointers, the types must match and not be `void *`. You can cast pointers to force compatibility. Example:

```
void function(void)
{
  char *char_ptr;
  int *int_ptr;
  void *void_ptr;
  int result;
  long difference;
  char_ptr = int_ptr;                 // NO
  char_ptr = (char *)int_ptr;         // OK
  void_ptr = int_ptr;                 // OK
  int_ptr = void_ptr;                 // OK
  result = (char_ptr == int_ptr);     // NO
  result = ((int *)char_ptr == int_ptr);
                                      // OK
  result = (char_ptr == (void *)int_ptr);
                                      // OK
  result = (void_ptr == int_ptr);     // OK
  difference = char_ptr - int_ptr;    // NO
  difference = char_ptr - (char *)int_ptr;
}                                     // OK
```

pragma parameter function prototype not found

You have a pragma parameter statement for a function but no prototype for it. The prototype must be come immediately after the pragma parameter statement, like this:

```
#pragma parameter Delay(__A0, __A1)
pascal void Delay(long num, long *final)
   = { 0xA03B, 0x2280 };
```

project is incompatible with this version of THINK Project Manager
THINK Project Manager can open projects created with only THINK C version 3.0 or later.

project is not 32-bit compatible
You're trying to run your application under the THINK Project Manager with System 7 in 32-bit mode, but you didn't set the 32-bit compatible flag in your application. To set that flag, use the **SIZE Flags** menu in the **Set Project Type** dialog.

project requires a larger memory partition
You need to increase the project's partition size. To set the partition size, use the Partition box in the **Set Project Type** dialog.

prototype required
prototype required for function *'symbol'*
When the Require prototypes option is on, you must provide a function prototype for each of your functions. You may have defined a prototype, but forgot to put a semicolon (`;`) at the end. Also, in a `class` declaration a prototype is required for `operator new` and `operator delete`, regardless of the setting of the Require prototypes option.

redundant application of 'const'
redundant application of 'pascal'
redundant application of 'volatile'
You repeated one of the qualifiers `const`, `pascal`, or `volatile` in a type declaration.

resource file *'filename'* not found
Standard Macintosh system error. This can occur whenever THINK Project Manager is trying to operate on a resource file such as, building a CODE resource with Merge option on, running or building an application).

resource not found
The Resource Copier couldn't find a resource that was listed in the file's resource map. This is probably due to a corrupted resource map. Try opening the resource file with ResEdit's Verify command to repair the file.

resource too big
See the error message "link failed" described earlier in this appendix.

resource type has already been declared

THINK Rez. THINK Rez allows you to decide if you want to allow redeclaration of resource types. This option can be set in the **THINK Rez options** dialog box. The error message is generated only when this option is turned off.

resource types have to be exactly four characters long

THINK Rez. All resource types must be exactly four characters long. The most common way to declare a resource type is to use single quotes around four characters.

```
'test'   //valid resource type
'tests'  //invalid resource type
```

second argument to function does not match prototype
second argument to function *'symbol'* does not match prototype

The second argument to a function does not match the prototype. The prototype may be explicit or inferred.

static function *'symbol'* referenced but not defined

You declared a function `static` and referred to it, but you did not define it in the same source file. Previous versions of THINK C treated these as if they were declared `extern`, but now it strictly enforces the declaration.

string too long (will be truncated)

THINK Rez, Warning. If no length indicator is given, a `pstring`, `wstring`, or `cstring` stores the number of characters in the corresponding data definition. However, the maximum size of a `pstring` is 255, and the maximum size of a `wstring` and `cstring` is 2,147,483,647.

'struct *symbol'* already defined

There is already a `struct` with that tag. Example:

```
typedef struct X { short x; };
typedef struct X { short y; };
```

'struct *symbol'* already in use

The `struct` tag is already used as a `struct` or `union` tag. Example:

```
typedef struct X { short a, b, c; };
typedef union X { short x; double y; };
```

switch value must be integral

A switch expression must be of type `char`, `int`, or `long` or an `unsigned` variant. Example:

```
char *p;
float f;
switch(p){…}     // NO: switch of pointer
switch(f){…}     // NO: switch of float
```

syntax error

There was a syntax error. These are some syntax common errors:

- Too many `}`s
- A label without `:`
- Malformed expressions
- Mismatched parentheses
- A missing semicolon

the change statement is not supported

THINK Rez, Warning. Because THINK Rez is a translator in an integrated environment, the `change` statement is not supported. THINK Rez can only compile resource files and add resource contributions to the resource fork of a project, it cannot delete, change or otherwise modify the resource fork of the project except to add to it.

the delete statement is not supported

THINK Rez, Warning. Because THINK Rez is a translator in an integrated environment, the `delete` statement is not supported. THINK Rez can only compile resource files and add resource contributions to the resource fork of a project, it cannot delete, change or otherwise modify the resource fork of the project except to add to it.

The file *'filename'* does not belong to any SourceServer project

You are trying to use the **Check In File** or **Check Out File** command in the **SourceServer** menu to operate on a file that doesn't belong to a ProjectorDB. If you want to associate this file with a ProjectorDB (check it in as a new file), use the **Check In** command.

The file *'filename'* has not been modified.

You are trying to use the **Check In File** command in the **SourceServer** menu to check in a read-only file.

The file *'filename'* is already modifiable.

You are trying to use the **Check Out File** command in the **SourceServer** menu to check out a file that is already checked out as modifiable, or is modified read-only.

The project document exceeds the 16MB limit.
Try using the "Store debug info separately" option.

This error occurs when the project is too large. The THINK Project Manager will refuse to work with it. Turn on the Store debug info separately option, or turn off debugging.

the resource type 'type' has been redeclared

THINK Rez, Warning. THINK Rez allows you to decide if you want to allow redeclaration of resource types. This option can be set in the **THINK Rez** options dialog box. You will get this warning when the option is enabled, otherwise THINK Rez will generate a fatal error and halt compilation of the file.

There are no SourceServer projects mounted.

You get this message when you choose **Check In**, **Check Out**, **Name Revision**, **Set ProjectorDB**, or **Unmount ProjectorDB** from the **SourceServer** menu when there are no ProjectorDB files mounted in SourceServer. You must first mount an existing ProjectorDB or create a new one with the **New ProjectorDB** or **Mount ProjectorDB** command.

third argument to function does not match prototype
third argument to function *'symbol'* does not match prototype

The third argument to a function does not match the prototype. The prototype may be explicit or inferred.

too many arguments

THINK C allows you to have up to 31 arguments in a function definition. Example:

```
f(arg1, arg2, …, arg31, arg32)
                        // one too many args
}
```

too many initializers

The number of initialization values exceeds the expected number of data items specified in the declaration of the data structure. Example:

```
char *directions[4] = {"north", "east",
      "south", "west", "lost" };    // NO
struct { short a,b,c; } x[2] =
      { 1, 2, {3, 4 } };            // NO
```

too many parameters to $$Format statement

THINK Rez. The `$$Format` function has a maximum of 20 parameters.

too many segments

Applications can have no more than 254 segments. Multi-segment desk accessories, device drivers, and code resources are limited to 30 segments.

ToolServer error

ToolServer couldn't execute the command you gave it.

undefined label *'symbol'*

You tried to jump to the label *symbol*, but you haven't defined that label within the function.

undefined '*symbol*'

See the error message "link failed" described earlier in this appendix.

unexpected end-of-file

THINK C reached the end of a source file before a C language construct was completed. Example:

```
main(
   // End of file encountered
   // before the close parenthesis.
```

unexpected end of file

THINK Rez. End-of-file was reached before a THINK Rez construct was completed. THINK Rez expected more resource or type statements. A common error is a missing `}`. Example:

```
type 'test'{
/* EOF - end of file before close brace */
```

unexpected end of resource

THINK Rez. THINK Rez expected to see at least one more statement before the closing '}'

unexpected end of data statement

THINK Rez. THINK Rez expected more resource statements to match the type statements, or you might be missing a }.

'union *symbol*' already defined

There is already a `union` with that tag. Example:

```
typedef union X { short x; double y; };
typedef union X { short y; double x; };
```

'union *symbol*' already in use

The `union` tag is already used as a `struct` or `union` tag. Example:

```
typedef union X { short x; double y; };
typedef struct X { short a, b, c; };
```

unknown class *'symbol'*

You use the identifier *symbol* as a class name but it is not defined or is not a class. Example:

```
short C;
void function(void)
{
   void *p = new A;  // NO: A not defined
   C::f();           // NO: C not a class
}
```

unknown CPU

Assembly. You specified a CPU in an `asm` statement that was not one of the recognized CPUs. Example:

```
asm 68010 {    ; NO: Only 68000, 68020,
   rtd   #8    ;     68030, 68040, 68881,
}              //    and 68882 are
               //    permitted.
```

unknown enumeration *'symbol'*

An enumeration declaration refers to an enumeration tag that you have not defined. Example:

```
enum unknown colors;    // NO
```

unknown enumerated value or label
THINK Rez. THINK Rez encountered an identifier that either had not been defined. Make sure that you are not missing a ‘,’ or ‘;’.

unknown error, ID=*x*
Macintosh system error *x* has occurred.

unknown instruction
Assembly. The inline assembler doesn’t recognize the mnemonic you used. Usually, you forgot the period in a mnemonic. Example:

```
asm {
   movew    d0,d1    ;NO
   move.w   d0,d1    ;OK
}
```

unknown resource type ‘symbol’
THINK Rez. All resource types must be defined before being referenced. Make sure that the resource type falls within the range for the given type definition, and that you have included the correct header file if it is a predefined resource type.

unknown string prefix ‘symbol’
THINK Rez. The numeric types (`bitstring`, `byte`, `integer` and `longint`) are fully specified like this:

```
[unsigned] [radix] numeric-type
      [= expr | symbol definition …];
```

Where radix is one of the following string constants:

```
hex   decimal  octal    binary   decimal
```

If any string constant other than the word unsigned or one of the radix given above appears before the numeric type, it is considered an unknown string prefix.

unknown struct *‘symbol’*
unknown union *‘symbol’*
You must define a `struct` or `union` before declare an instance of it. However, you can declare a pointer to an undefined `struct` or

`union` since THINK C knows the size of a pointer and doesn't need to know the size of the undefined `struct` or `union`. Example:

```
struct not_previously_defined s;    // NO
struct not_previously_defined *p;   // OK
short i = sizeof(p);                // OK
short j = sizeof(*p);               // NO
struct link_list_element {
  struct link_list_element *next;   // OK
  struct link_list_element recursive;
                                    // NO
};
```

unknown token

THINK Rez. Certain characters that are part of the ASCII character set are illegal tokens in THINK Rez. However, any character can occur within comments, string literals or character literals. Example:

```
integer = #define M 4;     /* Illegal */
```

unterminated comment

THINK C reached the end of a source file before it reached the end of a comment.

unterminated quote

Either a character constant or a string constant is missing its end quote. Example:

```
void function(void)
{
  long file_type;
  function("hello world);  // NO: missing "
                           //     after
                           //     world
  file_type = 'TEXT;       // NO: missing '
                           //     after
                           //     TEXT
  file_type = 'TEXT';      // OK
}
```

unterminated string

You left the closing quote off a string. Also check if you accidentally placed a lone quote mark in a file.

value has no members

In a `.` or `->` expression the left operand was not a struct or a union. Example:

```
void function(void)
{
  short non_struct, *short_ptr;
  struct s1_struct { short member_1; } s1,
*p1;

  non_struct.member_1;  // NO: non_struct
                        //     is not a
                        //     struct
  short_ptr->member_1;  // NO: short_ptr
}                       //     is not a
                        //     struct
                        //     pointer
```

'void' function must not return a value

A function declared `void` cannot return a value. Use a `return` statement without a value. Example:

```
void in_partners_suit(condition)
short condition;
{
   if (condition)
      return;        // OK
   else
      return(1);     // NO
}
```

volume is locked

You tried to use the **Save** or **Build** command to save a file to a hard drive or floppy disk that is locked, or to which you don't have write access; or, you tried to open a file located on a volume to which you don't have read access.

volume is locked

.o Converter. The .o Converter couldn't create the vocabulary file for the current object file because the volume that the `.o` file is on was locked. Either turn off the Create .v file option, unlock the disk that the `.o` file resides on, or move the `.o` file to a disk that's not locked.

volume not found (*'volumeName'*)

Standard Macintosh system error. This can occur whenever THINK Project Manager is searching for a file or folder (opening a project, saving a file, compiling, searching with the **Find** command, and so on).

wrong number of arguments to function
wrong number of arguments to function *'symbol'*

You called a function with an explicit or inferred prototype with the wrong number of arguments. To write a function that takes a variable number of arguments, use `...` in your prototype. Example:

```
typedef void (*actionProc)(char *s);
(*actionProc)(aString, anInt);        // NO

void error(char *s);
error("invalid command %s", aString); // NO

void msg(char *s, ...);
msg("invalid command %s", aString);   // OK
```

wrong number of arguments to macro *'symbol'*

You called a macro with the wrong number of arguments. Example:

```
#define twice(x)  (x+x)
void function(void)
{
  short i;
  i = twice(3,4);    // NO: macro called
}                    //     with 2 args.
```

wrong number of operands

Assembly. You forgot to supply a necessary operand. Example:

```
asm {
   add.w    d0       ; NO
   add.w    d0,d0    ; OK
}
```

wrong type(s) of operand(s)

Assembly. Some instructions require operands of a specific type. Example:

```
asm {
   movea    d1,d0    ; NO: MOVEA can move
}                    ;     only into
                     ;     address regs
```

wstring too long (will be truncated)

THINK Rez. If no length indicator is given, a `wstring` stores the length in the first two bytes of the string. The maximum size of a `wstring` is 65,535.

C Error Messages